Purge bath overflow sensor - DR

# Description

The purge bath overflow sensor is installed in a ~4 liter bath (or drip tray) positioned under the purge unit. It monitors the height of a float along the upper 8mm edge of the bath. In case of blockage or pump malfunction in the purge unit, excess waste material flows down to the bath. As the level of material inside the bath rises it will eventually carry the float up with it, causing the sensor to send a signal to the OCB, which will in turn be transferred to the EM.

According to Meir Bar Nathan, the purge rate peaks at 4gr/sec. Assuming 1gr = 1cm3 of material (actually our materials are denser than water), at this rate it will take 15.4 seconds for the waste level to climb 1mm up the bath which has a base area of 616cm2.

# SW environment

New elements are marked with a glow.

## EM: class relations

|  |
| --- |
| CActuatorBase |
| -Update…  -Get…  -PurgeBathSensorCalc  +GetActuatorOnOff  +TimedSetOnOff  +ActivateRollerSuctionSystem  +SetOnOff  +GetActuatorStatus  +GetInputStatus  +GetStabilizedPurgeBathSensorStatus |
| DbMutex  m\_ActuatorTable  m\_SensorTable  m\_PollingControFlag  m\_FlagCanSendGetStatusMsg |

|  |
| --- |
| CActuator |
| +GetReadResponse  -…AckResponse |
| m\_Flag…UnderUse |

|  |
| --- |
| CActuatorDummy |
|  |
|  |

|  |
| --- |
| COCBStatusSender |
|  |
|  |

|  |
| --- |
| MachineSequencer |
|  |
|  |

|  |
| --- |
| CErrorHandlerClient |
|  |
|  |

|  |
| --- |
| COCBProtocolClient |
|  |
|  |

|  |
| --- |
| CAppParams |
|  |
|  |

|  |
| --- |
| CBackEndInterface |
|  |
|  |

|  |
| --- |
| CSensorStabilizer |
| +UpdateHistory  +GetHistoryLastStatus |
|  |

|  |
| --- |
| CFixedSizeQueue |
|  |
|  |

## EM: Class interaction at init time

MainUnit

CQ2RT  
Application

Machine  
Manager

Machine  
Sequencer

CActuator

FormCreate

AppInit

Init

COCBStatus  
Sender

CActuator

## EM: Status checking

GetStabilizedPurgeBathSensorStatus

GetStabilizedPurgeBathSensorStatus

Machine  
Manager

Machine  
Sequencer

CActuator

PrePrintPhase

PrePrintSequence

PrintingPhase

EMachineSequencer(Q2RT\_PURGE\_BATH\_OVERFLOW)

PrintingSequence

EMachineSequencer(Q2RT\_PURGE\_BATH\_OVERFLOW)

## Overall: Class interaction during normal operation

GetInputs  
Task

CActuator

CFrontEnd  
Interface

COCBStatus  
Sender

CEdenProtocol  
Engine

CSerialSender  
Thread

CSerialReceiver  
Thread

EdenProtocol  
Decode

UART  
Int

Msg  
Decode

GetInputStatus

OCB\_GET\_INPUTS\_STATUS

InstallReceive  
Handler

EM

BuildRaw  
Packet

Send

COCBProtocolClient

Queue  
Outgoing

WaitReplyTimeout

🕐

Receive

Write

message

GetReading

EdenProtocolSend

getParams  
+resume

buffer

message

ProcessIncomingFullPacket

ProcessIncomingPacket

Timestamp

InputStatusAckResponse

UpdateInputTable

PurgeBathSensorCalc

FE\_CURRENT\_OCB\_SENSOR\_STATUS\_START+SENSOR\_ID\_xx

OCB

NotificationMessage("Purge bath overflow detected.\r\nPrinter maintenance required.")

GetInputs  
Task

CActuator

CFrontEnd  
Interface

COCBStatus  
Sender

CEdenProtocol  
Engine

CSerialSender  
Thread

CSerialReceiver  
Thread

EdenProtocol  
Decode

UART  
Int

Msg  
Decode

GetInputStatus

OCB\_GET\_INPUTS\_STATUS

InstallReceive  
Handler

EM

BuildRaw  
Packet

Send

COHDB  
ProtocolClient

Queue  
Outgoing

WaitReplyTimeout

🕐

Receive

Write

message

GetReading

EdenProtocolSend

getParams  
+resume

buffer

message

ProcessIncomingFullPacket

ProcessIncomingPacket

Timestamp

InputStatusAckResponse

UpdateInputTable

PurgeBathSensorCalc

FE\_CURRENT\_OCB\_SENSOR\_STATUS\_START+SENSOR\_ID\_xx

OHDB

NotificationMessage("Purge bath overflow detected.\r\nPrinter maintenance required.")

# Changes to the code

1. OCB simulator float sensor GUI – thanks to Luda.
2. EM⬄OCB messages – using the existing OCB\_GET\_INPUTS\_STATUS/GET\_INPUTS\_STATUS\_MSG and its OCB handler task.
3. HW access  
   - Sensors.c: SensorsInit() - OBJET\_MACHINE => P1.7 configured as analog input, else digital  
   - redefine IN\_SPARE\_1\_SENSOR\_ID as PURGE\_BATH\_OVERFLOW\_SENSOR for the first entry in SensorPortsLookup[] (value = PORT\_1) and SensorMaskLookup[] (value = SENSOR\_1\_MASK).



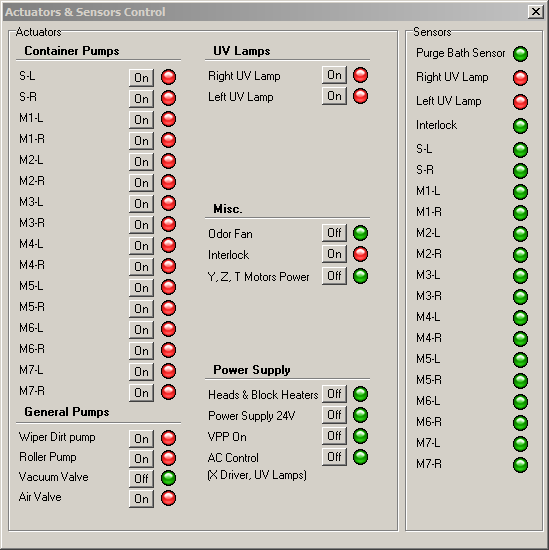
1. EM: Purge bath overflow sensor stability:

Requirement: sensor must reflect the status of the last ~30sec

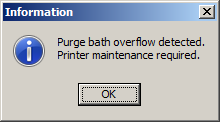
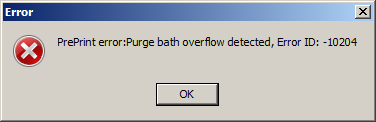
Implelemtation:

* OCB\_GET\_INPUTS\_STATUS refresh rate is ~2 sec => put through a sleeve of 15 samples. The sleeve width is configurable using the parameter *PurgeBathStabilizationWidth*.
* CSensorStabilizer implements hysteresis: while sensor status is “normal” sleeve should have 15 “overflow” samples to move sensor to “overflow”. While sensor status is “overflow” sleeve should have 15 “normal” samples to move to “normal”. The memory of the last 15 samples is inherited from CFixedSizeQueue.

1. Dialog boxes
   1. Actuators & sensors control

  
Note that the LED will change color immediately, but the sequencer will change its behavior after the stabilization time.

* 1. User notifications

1. Upon detection: Q2RTErrorDefsList.h  
   ErrorHandlerClient: see 3.5.b – final text TBD by Zecharia  
   Stop printing (see bottom of 2.3)  
   Do not allow new print jobs through checks in PrePrint stage (see 3.8)
2. AppParams.h/cpp, q2rt.cfg, ParamsDefault.h
   1. “PurgeBathOverflowSensorEnabled” 1=true, 0=false.  
      Description: Detects overflow in the purge bath, indicating a blockage/malfunction in the purge unit
   2. PurgeBathStabilizationWidth – positive integer.  
      Description: Controls purge bath overflow sensor stabilization time.
3. PrePrint Check (CMachineSequencer::PrePrintSequence):  
   Implemented as one of the first checks, before any long action starts during which the operator might leave the machine and miss the error message.  
   throw EMachineSequencer (see top of 2.3 and 3.5.b). New error: Q2RT\_PURGE\_BATH\_OVERFLOW
4. Wizard integration - for MRW, HOW and shutdown: The purge bath sensor status shall be accessible in python by calling *Actuator.GetStabilizedPurgeBathSensorStatus()* which returns true when an overflow is detected and false otherwise (incl. sensor not installed).
5. QTP automatic testing: Currently down and about to be rewritten, but generally the sensors group in the Actuators & Sensors Control dialog box (3.7 above) are found by their text, so moving them around should not ruin existing tests.
6. Debug prints:
   1. Whenever the stabilization changes its value the following log message will be issued:  
      CActuatorBase::PurgeBathSensorCalc - purge bath OK: *a*->*b*  
      Where *a*->*b* can be 1->0 in case an overflow was reported or 0->1 in case the overflow was resolved.
   2. Whenever an overflow sample is obtained the following log message will be issued:  
      Purge Bath Overflow: read *a* (accumulated *b*/*c*, reporting *d*)  
      where *a* is the value read, *b* is the number of overflow samples accumulated, *c* is the number of samples needed to stabilize the reading and *d* is the currently stabilized value.